

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

## **IMAGES ARE BEST AVAILABLE COPY.**

As rescanning documents *will not* correct images,  
Please do not report the images to the  
Image Problem Mailbox.

#3  
2186  
2-12-01

XA-9375  
PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

Kazuo AISAKA et al.

Appln. No.: 09/688,360

Group Art Unit: 2186

Filed: October 12, 2000

For: CACHE MEMORY ALLOCATION METHOD

\* \* \*

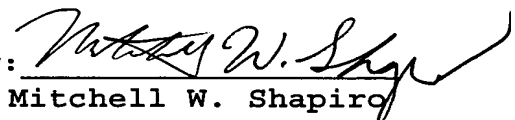
CLAIM OF PRIORITY UNDER 35 U.S.C. § 119

Assistant Commissioner for Patents  
Washington, D.C. 20231

Sir:

Applicants hereby claim the priority of Japanese Patent  
Application No. 11-291807 filed October 14, 1999 and submit  
herewith a certified copy of said application.

Respectfully submitted,

By:   
Mitchell W. Shapiro  
Reg. No. 31,568

MWS:dlb  
Vorys, Sater, Seymour  
and Pease LLP  
1828 L Street, N.W.  
Eleventh Floor  
Washington, D.C. 20036-5109  
Tel: (202) 467-8812  
January 26, 2001



RECEIVED  
JAN 29 2001  
Technology Center 2100

日本国特許庁  
PATENT OFFICE  
JAPANESE GOVERNMENT

31,701064 MC

09/688,360  
GAU 2186

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日

Date of Application:

1999年10月14日

出願番号

Application Number:

平成11年特許願第291807号

出願人

Applicant(s):

株式会社日立製作所



RECEIVED

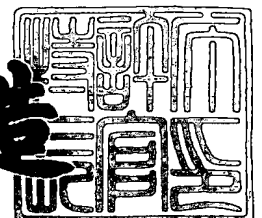
JAN 29 2001

Technology Center 2100

2000年10月 6日

特許庁長官  
Commissioner,  
Patent Office

及川耕造



出証番号 出証特2000-3081619

【書類名】 特許願

【整理番号】 H99010641A

【提出日】 平成11年10月14日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 12/08

【発明者】

    【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地  
株式会社日立製作所中央研究所内

    【氏名】 相坂 一夫

【発明者】

    【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地  
株式会社日立製作所中央研究所内

    【氏名】 十山 圭介

【特許出願人】

    【識別番号】 000005108

    【氏名又は名称】 株式会社日立製作所

【代理人】

    【識別番号】 100075096

    【弁理士】

    【氏名又は名称】 作田 康夫

    【電話番号】 03-3212-1111

【手数料の表示】

    【予納台帳番号】 013088

    【納付金額】 21,000円

【提出物件の目録】

    【物件名】 明細書 1

    【物件名】 図面 1

    【物件名】 要約書 1

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 キャッシュメモリの割当方法及びオペレーティングシステム及びそのオペレーティングシステムを有するコンピュータシステム

【特許請求の範囲】

【請求項 1】

複数のプログラム単位を実行するコンピュータシステムが有する資源の割当方法であって、

前記割当方法は、プログラム単位の実行要求を受付けるステップと、

前記プログラム単位の属性を示すパラメータを獲得するステップと、

前記パラメータ及び資源割当テーブル及びキャッシュ管理テーブルに基づいて、前記プログラム単位の実行に必要な資源を割当てるステップと、

前記資源割当の結果を前記資源割当テーブルに記憶するステップと、

前記プログラム単位が使用するキャッシュメモリの割当を前記キャッシュ管理テーブルに記憶するステップとを有していることを特徴とする資源の割当方法。

【請求項 2】

請求項 1 において、

前記パラメータは、前記プログラム単位の中で高頻度に行われる部分を示すための主要部位置情報を有し、

前記資源割当テーブルは、前記プログラム単位が使用する主記憶のアドレス情報を有し、

前記キャッシュ管理テーブルは、前記キャッシュメモリのアドレスと前記プログラム単位との対応を示しており、

前記資源を割当てるステップでは、前記主要部位置情報と前記アドレス情報と前記キャッシュ管理テーブルに基づいて、前記プログラム単位の主要部が使用するエントリアドレスが重ならないように前記プログラム単位が使用する前記主記憶のアドレスを決定することを特徴とする資源の割当方法。

【請求項 3】

請求項 1 において、

前記資源割当テーブルは、前記プログラム単位が使用する主記憶のアドレス情

報を有し、

前記キャッシュ管理テーブルは、前記キャッシュメモリのページと前記プログラム単位との対応を示しており、

前記資源を割当てするステップでは、前記パラメータと前記アドレス情報と前記キャッシュ管理テーブルに基づいて、前記プログラム単位が使用する前記主記憶のアドレスを決定することを特徴とする資源の割当方法。

【請求項 4】

複数のプログラム単位を実行する機能を有するオペレーティングシステムであって、

前記オペレーティングシステムは、新たなプログラム単位の実行要求を受付けると、

前記プログラム単位の属性を示すパラメータの獲得と、

前記パラメータ及び資源割当テーブル及びキャッシュ管理テーブルに基づいて、前記プログラム単位の実行に必要な資源の割当と、

前記資源の割当の結果の前記資源割当テーブルへの記憶と、

前記プログラム単位が使用するキャッシュメモリの割当の前記キャッシュ管理テーブルへの記憶とを行うことを特徴とするオペレーティングシステム。

【請求項 5】

請求項 4 において、

前記キャッシュメモリは、エントリアドレスを有し、

前記パラメータは、前記プログラム単位の中で高頻度に実行される部分を示す主要部位置情報を有し、

前記資源割当テーブルは、プログラム単位が使用する主記憶のアドレス情報を有し、

前記キャッシュ管理テーブルは、前記キャッシュメモリのアドレスとプログラム単位との対応情報を有し、

前記資源の割当とは、前記主要部位置情報及び前記アドレス情報及び前記キャッシュ管理テーブルに内容に基づいて、前記プログラム単位の中の前記高頻度に実行される部分に対応する前記エントリアドレスが前記プログラム単位以外のプロ

グラム単位の中の高頻度に行われる部分に対応するエントリアドレスと重複しないように、前記プログラム単位が使用する主記憶のアドレスを決定するものであることを特徴とするオペレーティングシステム。

【請求項 6】

請求項 4 において、  
前記キャッシュメモリは、複数のページを有し、  
前記資源割当テーブルは、プログラム単位が使用する主記憶のアドレス情報を有し、  
前記キャッシュ管理テーブルは、前記キャッシュメモリのページとプログラム単位との対応情報を有し、  
前記資源の割当とは、前記主要部位置情報及び前記アドレス情報及び前記キャッシュ管理テーブルに内容に基づいて、前記プログラム単位が使用する主記憶のアドレスを決定するものであることを特徴とするオペレーティングシステム。

【請求項 7】

請求項 6 において、  
前記プログラム単位から別のプログラム単位へ実行を移す際、前記キャッシュ管理テーブルに従ってキャッシュページレジスタの書換を行うことを特徴とするオペレーティングシステム。

【請求項 8】

請求項 6 または 7 において、  
前記資源の割当とは、前記パラメータが有する優先度情報に基づき、前記プログラム単位に割当てた前記キャッシュメモリのページの数を決する処理を含むものであることを特徴とするオペレーティングシステム。

【請求項 9】

請求項 6 または 7 において、  
前記資源の割当とは、前記パラメータが有する優先度情報に基づき、前記キャッシュメモリの所定のページに割当てたプログラム単位の数を決する処理を含むものであることを特徴とするオペレーティングシステム。

【請求項 1 0】

キャッシュメモリと、CPUと、メモリとを有するコンピュータシステムであって、

前記コンピュータシステムは、前記メモリに格納されたオペレーティングシステムによって制御され、

前記オペレーティングシステムは、前記コンピュータシステムがプログラム単位の実行要求を受付けると、

前記プログラム単位の属性を示すパラメータの獲得と、

前記パラメータ及び資源割当テーブル及びキャッシュ管理テーブルに基づいて、

前記プログラム単位の実行に必要な資源の割当と、

前記資源の割当の結果の前記資源割当テーブルへの記憶と、

前記プログラム単位が使用するキャッシュメモリの割当の前記キャッシュ管理テーブルへの記憶とを行うことを特徴とするコンピュータシステム。

【請求項 1 1】

請求項 1 0 において、

前記キャッシュメモリは、エントリアドレスを有し、

前記パラメータは、前記プログラム単位の中で高頻度に実行される部分を示す主要部位置情報を有し、

前記資源割当テーブルは、プログラム単位が使用する主記憶のアドレス情報を有し、

前記キャッシュ管理テーブルは、前記キャッシュメモリのアドレスとプログラム単位との対応情報を有し、

前記資源の割当とは、前記主要部位置情報及び前記アドレス情報及び前記キャッシュ管理テーブルに内容に基づいて、前記プログラム単位の中の前記高頻度に行われる部分に対応する前記エントリアドレスが前記プログラム単位以外のプログラム単位の中の高頻度に行われる部分に対応するエントリアドレスと重複しないように、前記プログラム単位が使用する主記憶のアドレスを決定するものであることを特徴とするコンピュータシステム。



【請求項 1 2】

請求項 1 0 において、  
 前記キャッシュメモリは、複数のページを有し、  
 前記資源割当テーブルは、プログラム単位が使用する主記憶のアドレス情報を有し、  
 前記キャッシュ管理テーブルは、前記キャッシュメモリのページとプログラム単位との対応情報を有し、  
 前記資源の割当とは、前記パラメータ及び前記アドレス情報及び前記キャッシュ管理テーブルに内容に基づいて、前記プログラム単位が使用する主記憶のアドレスを決定するものであることを特徴とするコンピュータシステム。

【請求項 1 3】

CPUとキャッシュメモリとメモリとを有し、複数のプログラム単位を実行するコンピュータシステムであって、  
 前記コンピュータシステムは、資源の割当結果を記憶する資源割当テーブルと前記キャッシュメモリの割当を記憶するキャッシュ管理テーブルとを有しており、  
 前記メモリは、前記コンピュータシステムを制御するためのオペレーティングシステムを格納する領域を有しており、  
 前記オペレーティングシステムは、前記複数のプログラム単位の内一つのプログラム単位を実行する際、前記一つのプログラム単位に関するパラメータを獲得し、前記パラメータ及び前記資源割当テーブル及び前記キャッシュ管理テーブルの内容に基づいて、前記一つのプログラム単位に対応する資源の割当を行うことを特徴とするコンピュータシステム。

【請求項 1 4】

請求項 1 3 において、  
 前記キャッシュメモリは、エントリアドレスを有し、  
 前記パラメータは、前記プログラム単位の中で高頻度に行われる部分を示す主要部位置情報を有し、  
 前記資源割当テーブルは、プログラム単位が使用する主記憶のアドレス情報を有し、

前記キャッシュ管理テーブルは、前記キャッシュメモリのアドレスとプログラム単位との対応情報を有し、

前記資源の割当とは、前記主要部位置情報及び前記アドレス情報及び前記キャッシュ管理テーブルに内容に基づいて、前記プログラム単位の中の前記高頻度に行われる部分に対応する前記エントリアドレスが前記プログラム単位以外のプログラム単位の中の高頻度に行われる部分に対応するエントリアドレスと重複しないように、前記プログラム単位が使用する主記憶のアドレスを決定するものであることを特徴とするコンピュータシステム。

【請求項 1 5】

請求項 1 3 において、

前記キャッシュメモリは、複数のページを有し、

前記資源割当テーブルは、プログラム単位が使用する主記憶のアドレス情報を有し、

前記キャッシュ管理テーブルは、前記キャッシュメモリのページとプログラム単位との対応情報を有し、

前記資源の割当とは、前記パラメータ及び前記アドレス情報及び前記キャッシュ管理テーブルの内容に基づいて、前記プログラム単位が使用する主記憶のアドレスを決定するものであることを特徴とするコンピュータシステム。

【請求項 1 6】

複数のプログラム単位を実行するコンピュータシステムを制御するオペレーティングシステムであって、

前記オペレーティングシステムは、前記コンピュータシステムが前記複数のプログラム単位の内の一つのプログラム単位を実行する際、前記一つのプログラム単位の主要部が使用するキャッシュメモリのエントリアドレスが、前記複数のプログラム単位の内の前記一つのプログラム単位とは異なるプログラム単位の主要部が使用する前記キャッシュメモリのエントリアドレスと重複しないように、前記一つのプログラム単位に対して資源を割当ててことを特徴とするオペレーティングシステム。

【請求項 1 7】

請求項 1 6 において、  
前記資源の割当は、前記オペレーティングシステムが有する資源割当手段によってなされることを特徴とするオペレーティングシステム。

【請求項 1 8】請求項 1 6 において、

前記オペレーティングシステムは、前記コンピュータシステムが有するメモリに格納されていることを特徴とするオペレーティングシステム。

【請求項 1 9】オペレーティングシステムによって制御され、複数のプログラム単位を実行するコンピュータシステムであって、

前記コンピュータシステムは、前記複数のプログラム単位の内の一つを実行する際、前記一つのプログラム単位の主要部が使用するキャッシュメモリのエントリアドレスが、前記複数のプログラム単位の内の前記一つのプログラム単位とは異なるプログラム単位の主要部が使用する前記キャッシュメモリのエントリアドレスと重複しないように、前記オペレーティングシステムに基づいて、前記一つのプログラム単位に対して資源を割当ててことを特徴とするコンピュータシステム。

【請求項 2 0】請求項 1 9 において、

前記資源の割当は、前記コンピュータシステムに存在する資源割当手段によってなされることを特徴とするコンピュータシステム。

【請求項 2 1】請求項 2 0 において、

前記資源割当手段は、前記オペレーティングシステムが有していることを特徴とするコンピュータシステム。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明はコンピュータシステムが有するキャッシュメモリの割当方法に関するものであり、更には、キャッシュメモリの割当機能を有するオペレーティングシステム及びそのオペレーティングシステムにより動作するコンピュータシステムに関する。

## 【0002】

## 【従来の技術】

マイクロプロセッサのハードウェアが進歩するにつれて、その上で動作するソフトウェアにも、従来に増して高度な機能が要求される様になりつつある。この一環として、1台のマイクロプロセッサで複数のソフトウェアを同時並行的に動作させる「マルチプロセス機能」が、パーソナルコンピュータ等の小型の情報処理装置においても要求される様になっている。

## 【0003】

この実現のためのソフトウェアとして、複数のプロセスを整理する役目を持つ「オペレーティングシステム（OS）」が用意され、複数のプロセスをメモリ空間に配置し、時分割方式で交互に実行させる。メモリ・入出力チャネル・CPU時間など、各プロセスが実行に必要な資源を配分するのもOSの役割である。いくつかのプロセスの間でそれら資源の奪い合い（競合）が生じた場合は、OSが適当な方法で調停を行なう。

## 【0004】

マイクロプロセッサの進歩を示すもう一方の側面として、処理速度の高速化があげられる。現在最先端のマイクロプロセッサは、約15年前の大型コンピュータを凌ぐ性能を持っており、当時は困難であった映像処理機能などがマイクロプロセッサを用いて実現されつつある。

## 【0005】

高速化の為のハードウェア技術としては、プロセッサに用いる回路素子自体を高速化する事は勿論であるが、キャッシュメモリの利用により、ハードウェアの平均的な処理速度を向上させる技術が広く用いられている。

## 【0006】

## 【発明が解決しようとする課題】

キャッシュメモリとは、通常の記憶装置（主記憶メモリ、または単に主記憶）に付け加えられて利用される少量の高速メモリの事を言う。キャッシュメモリの読出し／書込み時間は、主記憶の5～10倍程度高速である。この一方で主記憶メモリの記憶容量は、現在の標準的なパーソナルコンピュータにおいて主記憶が

1 6 ~ 6 4 M B であるのに対し、キャッシュメモリは 8 ~ 3 2 K B と小さい。

【 0 0 0 7 】

パーソナルコンピュータ等の C P U がプログラムを実行する際に、キャッシュメモリは以下の様に利用される。

【 0 0 0 8 】

電源投入直後はキャッシュメモリは空である。プログラムの実行が開始されると、C P U はプログラム中の命令を主記憶から 1 つずつ読み出して解釈・実行するが、この時並行して、読み出した命令をキャッシュメモリにコピーしておく。キャッシュメモリが満杯の場合は、過去にコピーした命令のうち適当なものを消して、現在の命令をコピー、つまり上書きする。

【 0 0 0 9 】

ここまでの過程では、キャッシュメモリを用いても処理速度は向上しない。しかしながら大半のプログラムでは、途中で処理の繰返しが起こり、プログラムにおいて過去に実行した部分を再度実行する状態が生じる。この場合 C P U は、主記憶にはアクセスせず、キャッシュメモリにコピーされた命令を読み出す。これにより、読み出しが 5 ~ 1 0 倍程度高速にできる。この効果は、繰返しの回数が多いほど大きい。

【 0 0 1 0 】

コンピュータのプログラムは、多くの応用分野において、比較的単純な処理を何回も繰返しを行なう事が多い。たとえばコンピュータグラフィックスのプログラムでは、「画面上のある点の明るさを特定の数式に従って決める」という処理を、画面全部の点について繰返す。また、会計処理のプログラムでは、「伝票内の全ての金額を合計する」という処理を、全ての伝票に対して繰返す。これらの様に、プログラム中の特定の部分が集中的に繰返されるという性質を「プログラムの局在性」と呼び、大半のプログラムがこの性質を持っている事が知られている。

【 0 0 1 1 】

つまり、キャッシュメモリによるプログラムの読み出し速度の高速化という効果を得るためには、繰返し 1 回分に相当するプログラムのある部分がキャッシュ

メモリにコピーされる必要があり、相応する記憶容量がキャッシュメモリに必要となる。実用のコンピュータにおいては、キャッシュメモリの容量は上記の必要性を満足するように決定されており、パーソナルコンピュータのCPUとして用いられるマイクロプロセッサの場合は8～32KB程度である。

## 【0012】

また、以上の動作においては、CPUはプログラムの命令を読出す際に、当該命令がキャッシュメモリにコピーされている（キャッシュヒット）かされていない（キャッシュミス）かどうかを判定する必要がある。これは従来、以下の方法で行なわれる事が多い。

## 【0013】

キャッシュメモリはメモリの一種であるので、各記憶単位（バイト）にアドレスが付されている。たとえば、キャッシュメモリの容量が8KB（2\*\*13バイト、但し\*\*はべき乗を表す）であるならば、各バイトには2進数の“000000000000”から“111111111111”まで、13bitのアドレスが付されている。一方、主記憶にも同様の方法でアドレスが付されている。たとえば、主記憶の容量が32MB（2\*\*25バイト）であるならば、各バイトには25bitのアドレスが付されている。

## 【0014】

CPUがキャッシュメモリに命令をコピーする際には、主記憶のアドレス信号線をそのまま流用してキャッシュメモリのアドレスとする。但しキャッシュメモリの方が容量が小さい（アドレス信号線が少ない）ので、主記憶のアドレス信号線のうち下位側の相当数を利用する。例えば上述の容量のメモリにおいて、主記憶のアドレス

$$A = A_{24} \ A_{23} \ A_{22} \ \dots \ A_{13} \ A_{12} \ A_{11} \ \dots \ A_2 \ A_1 \ A_0$$

（各 $A_n$ は0又は1）

にある命令は、キャッシュメモリのアドレス

$$A = A_{12} \ A_{11} \ \dots \ A_2 \ A_1 \ A_0$$

にコピーされる。これを数式で表現すると

$$A = A \bmod (2^{**13}) \quad \text{但し mod は剰余をとる演算を表す} \dots \dots \text{[式 1]}$$

」という関係となる。この方法では、主記憶のアドレス  $A$  が与えられれば、対応するキャッシュメモリのアドレス  $A'$  は一意的に決定される。

【0015】

但しこれだけでは、 $A'$  が与えられても、その内容が主記憶のどの番地のコピーであるかは、 $A$  の上位側のビット  $A_{24} \sim A_{13}$  の部分がわからないため決定できない。そこでキャッシュメモリに命令をコピーする際には、上記の

$$A_t = A_{24} \ A_{23} \ A_{22} \ \dots \ A_{13}$$

を命令と合せてキャッシュメモリの  $A'$  番地に記憶しておく。このためキャッシュメモリには、各バイトに対して 12 (= 25 - 13) bit ずつの補助的な記憶容量を設けておく。上記  $A_t$  は、主記憶のアドレス  $A$  に対する「タグ」と呼ばれる。

【0016】

なお上記構成は、最も単純なキャッシュメモリ構成を説明したものであるが、この構成では 1 バイト (8 bit) の命令を記憶するために 12 bit のタグを合せて記憶せねばならず、効率が悪い。そこで上記の改良として、ライン構成のキャッシュメモリが普及している。ラインとはキャッシュメモリの連続した複数バイト (たとえば 4 バイト) をひとまとめにした単位であり、命令のコピーやキャッシュヒット/ミスの判定はライン単位で行なう。ライン構成の場合、タグは各ラインに対して 1 つずつ記憶すれば良いので、記憶容量が節約できる。各ラインにはアドレスが付されるが、このアドレスはバイト単位の通常のアドレスとは意味が異なるので、エントリアドレス (又は単にエントリ) と呼ばれる。

【0017】

ライン構成のキャッシュメモリにおいても、主記憶のアドレスとキャッシュメモリのアドレス (エントリアドレスではなくバイト単位のアドレス) との対応関係が [式 1] で定められる事、プログラムの局在性がキャッシュメモリを有効に使うための前提条件となること、等の性質は変らない。

【0018】

勿論、キャッシュメモリにコピーされるのはプログラム中の命令に限定されることなくプログラムが扱うデータであっても問題はない。

## 【0019】

以上の様に、キャッシュメモリはコンピュータの性能向上に役立っている。しかしながら、コンピュータをマルチプロセスで使用する際には、キャッシュメモリが必ずしも効果的に機能しない場合がある事が近年わかってきた。

## 【0020】

キャッシュメモリは前項で述べた通り、プログラム（およびデータ）の局在性を前提とした性能向上方式である。一方マルチプロセスは、複数のプログラムを時分割で同時並行的に処理する方式である。従ってマルチプロセスでは、一定の時間周期で、全く異なったプログラムへ処理が移行する。これは本質的にプログラムの局在性を損なう原因となる。すなわち、移行の直後は、キャッシュメモリには移行前のプログラムがコピーされた状態になっているので、移行後のプログラムは主記憶から命令を読込まねばならず、キャッシュメモリの効果が得られない。その後、処理の繰返しが生じれば、キャッシュメモリが有効に働く事になる（各々のプログラムは、個別には十分な局在性を持っているものとする）が、その効果も次の移行までに限られる。

## 【0021】

この現象が特に著しいのは、マルチプロセスにおける時分割の単位時間（タイムスライス）が短く、その間にプログラムが少回数（5回程度）の繰返ししか処理できない場合である。この場合、例えばキャッシュメモリが主記憶に比べ10倍高速であるとしても、最初の1回は主記憶にアクセスせねばならないので、全体の読出速度は約3.5倍しか高速化しない。これは、十分な回数の繰返しではほぼ10倍の高速化が得られる事と比較すると、読出速度を約1/3に低下させる結果となる。

## 【0022】

上記の様な、キャッシュメモリへの上書きが頻発するために性能が低下する現象は、スラッシングと呼ばれている。スラッシングを回避するために、以下の通りいくつかの改善策が提案されているが、いずれも十分な効果をあげるに至っていない。



## 【 0 0 2 3 】

改善策の第 1 として、特定のプロセス（優先プロセス）のみにキャッシュメモリの利用を許し、他のプロセスは命令をキャッシュメモリにコピーしない、という方式がある。この方式は、優先プロセスに対してはシングルプロセスの場合と同等の効果を得る事ができる。しかしながら他のプロセスに対しては、キャッシュメモリを備えない低性能の処理を強いる事になる。

## 【 0 0 2 4 】

改善策の第 2 として、キャッシュメモリを並列化する事により、プロセス間でキャッシュの奪い合いが生じない様にする方式がある。すなわち、キャッシュメモリを複数組（たとえば 2 組）設け、命令をコピーする際にはいずれか空いている方の組にコピーする。キャッシュヒット／ミスの判定では、主記憶アドレスからエントリアドレスを作成した後、当該エントリアドレスに記憶されたタグを、2 つの組両方で主記憶アドレスの上位ビットと比較し、どちらか一方が一致すればその組にコピーされている内容を利用する。両方共不一致の場合はキャッシュミスとなり、主記憶にアクセスする必要が生じる。

## 【 0 0 2 5 】

この方法は、プロセスの数が組の数以下であればスラッシングを回避できる。しかしながらプロセスの数が組の数を上回ると、前述と同様のスラッシングが生じる。近年のパソコンでは、多数（5 個以上）のプロセスが同時に実行され、しかも各プロセスが更に複数のサブプロセス（スレッド）に分割される事が多く、これに対応できる様に十分な組数のキャッシュメモリを設ける事は困難である。また、仮に十分な組数を設けたとしても、それらを並列に動作させるためには複雑な制御回路が必要になる。例えば上述のキャッシュヒット／ミスの判定では、各組毎にタグの比較を行なった後、比較結果に従って特定の組を選択する機能が必要となる。特にキャッシュミスの場合は、主記憶から読み出した命令をどの組へ上書きするかを決定する為に複雑な計算（LRU 算法など）が必要となり、処理時間が増大する。この結果、キャッシュメモリの特長であるべき高速性が損なわれる事になり、実用的でない。現在広く使われているパソコン用 CPU においては、4 組（4 ウェイと呼ばれる）～ 8 組程度が上限となっている。

## 【 0 0 2 6 】

本発明の目的は、上記の状況を改善し、ハードウェアを極力増加させずにスラッシングを回避できるキャッシュメモリの使用方法を提供する事にある。

## 【 0 0 2 7 】

本発明のその他の目的については、本明細書及び図面より明らかになるであろう。

## 【 0 0 2 8 】

## 【課題を解決するための手段】

本願において開示される発明のうち代表的なものの概要を以下に示す。

## 【 0 0 2 9 】

すなわち、複数のプログラム単位を実行するコンピュータシステムが有する資源の割当を、プログラム単位を実行する要求を受付けるステップと、前記プログラム単位の属性を示すパラメータを獲得するステップと、前記パラメータ及び資源割当テーブル及びキャッシュ管理テーブルに基づいて前記プログラム単位の実行に必要な資源を割当てするステップと、前記資源割当の結果を前記資源割当テーブルに記憶するステップと、前記プログラム単位が使用する資源の割当を前記キャッシュ管理テーブルに記憶するステップとにより行う。

## 【 0 0 3 0 】

更に、具体的には、前記パラメータは前記プログラム単位の中で高頻度に実行される部分を示すための主要部位置情報を有し、前記資源割当テーブルは前記プログラム単位が使用する主記憶のアドレス情報を有し、前記キャッシュ管理テーブルは前記キャッシュメモリのアドレスと前記プログラム単位との対応を示しており、前記資源を割当てするステップでは、前記主要部位置情報と前記アドレス情報と前記キャッシュ管理テーブルに基づいて、前記プログラム単位の主要部が使用するエントリアドレスが重ならないように前記プログラム単位が使用する前記主記憶のアドレスを決定することにより、資源の割当を行う。

## 【 0 0 3 1 】

また、複数のプログラム単位を実行する機能を有するオペレーティングシステムが、新たなプログラム単位の実行要求を受付けると、前記プログラム単位の属

性を示すパラメータの獲得と、前記パラメータ及び資源割当テーブル及びキャッシュ管理テーブルに基づいて前記プログラム単位の実行に必要な資源の割当と、前記資源の割当の結果の前記資源割当テーブルへの記憶と、前記プログラム単位が使用するキャッシュメモリの割当の前記キャッシュ管理テーブルへの記憶とを行う。

【 0 0 3 2 】

更に、前記キャッシュメモリが複数のページを有している場合、前記資源割当テーブルは、プログラム単位が使用する主記憶のアドレス情報を有し、前記キャッシュ管理テーブルは、前記キャッシュメモリのページとプログラム単位との対応情報を有し、前記資源の割当とは、前記主要部位置情報及び前記アドレス情報及び前記キャッシュ管理テーブルに内容に基づいて、前記プログラム単位が使用する主記憶のアドレスを決定するものである。

【 0 0 3 3 】

また、キャッシュメモリと、CPUと、メモリとを有し、前記メモリに格納されたオペレーティングシステムによって制御されるコンピュータシステムは、新たなプログラム単位の実行要求を受付けると、前記プログラム単位の属性を示すパラメータの獲得と、前記パラメータ及び資源割当テーブル及びキャッシュ管理テーブルに基づいて、前記プログラム単位の実行に必要な資源の割当と、前記資源の割当の結果の前記資源割当テーブルへの記憶と、前記プログラム単位が使用するキャッシュメモリの割当の前記キャッシュ管理テーブルへの記憶とを行う。

【 0 0 3 4 】

以上に示したの構成により、前述した課題を解決することが可能となる。スラッシングの大きな要因は複数のプログラム単位（プロセス又はスレッド）がキャッシュメモリを奪い合う事にあり、これを回避するには、従来行なわれてきたハードウェアのみの改善では限界がある。従って本発明では、ソフト／ハードが協調した解決策を提案する。すなわち本発明においては、ハードウェア規模に制約がある限りキャッシュメモリを奪い合う現象はある程度避けられない、という前提の下に、それが著しいスラッシングに発展しないための方策を、ソフトウェア側の工夫によって提供する。

## 【 0 0 3 5 】

キャッシュミスによる影響が大きいのは、プログラムの中で高頻度に実行される部分（主要部）でスラッシングが生じる場合である。ところが、主要部の大きさはあまり大きくない事が知られている。そこで本発明では、マルチプロセス環境において、各々のプロセスの主要部が互いに異なるキャッシュメモリのアドレスを使用する様に、プロセスの主記憶上での配置を調整する方法を提供する。更に、前記調整を、プロセスにメモリその他の資源を割付ける役目を持つ上位のソフトウェア、すなわちオペレーティングシステム（OS）または相応の管理ソフトに行わせる。OS上で動作するアプリケーションプログラムは、OSが調整を可能とするための情報（パラメータ）をOSに提供する。OSの内部には、従来から用いられてきた資源管理テーブルに加えて、キャッシュ管理テーブルを設け、上記の調整に用いる。

## 【 0 0 3 6 】

以上の発明は、ソフトウェア単独で用いても有効であるが、ハードウェア側で用意されているスラッシング回避方策（キャッシュメモリをページに分割する、等）と併用すると更に効果的である。

## 【 0 0 3 7 】

## 【発明の実施の形態】

以下、本発明の実施例を図面を用いて説明する。

## 【 0 0 3 8 】

まず、本発明の実施対象となるコンピュータシステムの全体の構成を図2を用いて説明する。

## 【 0 0 3 9 】

コンピュータシステム10000にはハードウェアとして、マイクロプロセッサを用いたCPU1000、ROM1001、RAM1002が備えられ、これらがバス1003で接続されている。更にバス1003には入出力チャンネル1004が接続され、これを介して、大容量の補助記憶装置1005、オペレータが操作を入力するための入力装置1006、プログラムの実行結果をオペレータに知らせる為の表示装置1007が接続される。入力装置としてはキーボード、マ

ウス等が、表示装置としてはCRTディスプレイ、プリンタ等が通常用いられる。補助記憶装置1005としては、ハードディスク装置、フロッピーディスク装置などが通常用いられる。

#### 【0040】

CPUの内部には、システムの高速化を図るためのキャッシュメモリシステム100が設けられる。キャッシュメモリの具体例を、図5を用いて説明する。

#### 【0041】

図5はダイレクトマッピング方式と呼ばれる命令用キャッシュメモリシステムのハードウェア構成であり、キャッシュサイズ8192(8k)バイト、1ライン4バイト(従ってエントリアドレスは\$000~\$7FF、但し\$は16進数を表す記号、となる)の場合の例である。また主記憶容量は32MB(アドレス25bit)でバイト単位にアドレスが付されているものとする。主記憶とCPU間での命令の転送は、1バイト(8bit)幅のデータバスを通じて行なわれるものとする。

#### 【0042】

CPU1000内のキャッシュメモリシステム100は、以下の構成要素を有する。

#### 【0043】

まず、命令のコピーを記憶するための記憶手段150が設けられる。記憶手段150はVビット群151、タグ群152、ライン群153からなる。Vビット群151は、個々のエントリアドレスに対応するVビット(有効ビット)111をエントリアドレスの数だけ並べたものである。タグ群152は、個々のエントリアドレスに対応するタグ112をエントリアドレスの数だけ並べたものである。ライン群153は、個々のエントリアドレスに対応するライン113をエントリアドレスの数だけ並べたものであり、命令のコピーはここに記憶される。即ち、キャッシュメモリの本質的な機能を受け持つ部分である。以上の記憶手段150には、高速の記憶素子が用いられる。

#### 【0044】

アドレス選択回路101は、エントリアドレスを入力として受け取り、Vビッ

ト群 151、タグ群 152、ライン群 153 の各々から、当該エントリアドレスに対応する V ビット 111、タグ 112、ライン 113 を選択する。選択された V ビット 111、タグ 112、ライン 113 は、以後のキャッシュメモリシステムの動作において読出／書込操作の対象となる。

#### 【0045】

またキャッシュメモリシステム 100 には、タグを比較するための比較回路 103、キャッシュヒット／ミスの区別を示すヒット信号 104、ラインの中から所望のバイトを選択するバイトセクタ 105、キャッシュミス時に用いるキャッシュフィルカウンタ 106 および加算器 107、バイトセクタ 105 が使用するアドレスを選択するためのアドレスセクタ 108、等が設けられ、図示の通りに接続されている。

#### 【0046】

以上の回路は以下の通り動作する。

#### 【0047】

まず、電源投入（又はシステムリセット）時には、リセット信号 126 により全てのエントリアドレスに対する V ビットを 0 にクリアして、キャッシュの内容を無効とする。

#### 【0048】

その後、CPU 1000 がプログラムカウンタ（PC）1100 で指定される主記憶アドレス A の命令を実行しようとする時、まずキャッシュメモリの対応するアドレスに当該命令がコピーされているかどうかを調べる。このためにまず、PC の第 2 ～ 12 ビット、つまりエントリアドレスをアドレス選択回路 101 に入力し、対応するラインのタグ 112 および V ビット 111 を求める。タグ 112 を比較器 103 により A の上位（第 13 ～ 24）ビットと比較し、両者が一致かつ V ビットが 1（有効）であれば、当該ラインに所望の命令がコピーされている事示すヒット信号 104 が 1 となる。この場合、当該ライン 113 をライン群 153 から読出した後、A の第 0 ～ 1 b i t に従ってバイトセクタ 105 により所望のバイトを選択する。以上により選択されたバイトの内容が所望の命令であり、CPU 内の命令解読回路 1200 へと入力され、以降の命令実行に供さ

れる。

#### 【0049】

ヒット信号104が0の場合はキャッシュミスであり、主記憶から命令を読み出してキャッシュにコピーし直す（キャッシュフィル）必要がある。キャッシュフィルの手順は本発明の実施に影響を及ぼさないので、以下に概要のみを説明する。

#### 【0050】

本キャッシュメモリはライン構成なので、キャッシュフィルの時には主記憶アドレスAのみならず、同アドレスを含む1ライン（4バイト）分の命令をキャッシュメモリにコピーする。このためには、キャッシュフィルカウンタ106を用いて、当該アドレスと上位23ビット（第2～24bit）が同一となる4つのアドレスを逐次作り出し、アドレスバス121を通じて主記憶メモリにアクセスし、データバス122を通じて命令を取得する。取得された命令は、バイトセレクタ105により当該ライン内の対応するバイトに振り分けて記憶（コピー）される。以上が完了した後、当該ラインのVビット111をキャッシュフィル完了信号125により1にセットする。加算器107は、キャッシュフィルカウンタ106を0→1→2→3と変化させて4つの命令を順次取得する際に、最初に取得する命令が主記憶アドレスAの命令となるために設けられる。またキャッシュフィルが行なわれることを主記憶に知らせるため、ヒット信号104の論理否定であるキャッシュミス信号120が主記憶へ伝達される。以上の構成のキャッシュメモリによって、プログラムの局在性が成り立つ場合には、ハードウェアの平均的な処理速度を向上させる事が出来る。

#### 【0051】

また図2に示すCPUにはタイマ1010が接続されており、一定時間（たとえば10ミリ秒）毎にタイマ割込信号線1011を通じてCPUに割込みをかける。

#### 【0052】

以上のコンピュータシステムを動作させるソフトウェアとしては以下のものが用意されている。

## 【0053】

特に限定される訳ではないが、ROM1001内に、オペレーティングシステム(OS)200が用意され、本コンピュータシステム全体の動作を制御する。OSは後述の方法により、複数のプログラム単位を時分割で並行して実行するマルチプロセス機能を有し、このための資源割当手段999を内包する。またOSは自身が動作するための作業領域として、RAM内にOS領域210を確保し、占有的に使用する。ハードウェアのうち、入出力チャンネル1004、及びこれに接続される補助記憶装置1005、入力装置1006、出力装置1007はOSによりソフトウェア的に制御される。

## 【0054】

またOSがタイマ割込信号に応じて所定の動作をするため、ROM内の規定のアドレスに、OS内の割込処理機能プログラムの開始番地である割込アドレス201が記録されている。タイマ割込信号線1011を通じて割込信号が到来した時には、CPU内部の専用回路(図示せず)が動作して割込アドレス201を参照し、同アドレスに分岐する。

## 【0055】

ユーザが使用するアプリケーションプログラム221、222、……、は補助記憶装置1005に格納されており、必要に応じてオペレータの指示操作の下に、RAM1002に複写されて実行される。各アプリケーションプログラムには、当該プログラムの実行に必要な属性情報231、232、……、が付随させており、OSにて利用する。属性情報の内容は、当該プログラムの大きさ、実行開始番地、必要な作業メモリ量などである。

## 【0056】

以上で述べたシステムにおいて、OSの動作は図3及び図4の通りとなる。

## 【0057】

まず電源投入(又はシステムリセット)時に行なう準備動作を、図3のフローチャートを用いて説明する。

## 【0058】

OSの動作準備完了前に割込が入るのを防ぐために、割込禁止命令を発行する



(ステップ 3 0 1、以下、簡略のため s 3 0 1 と記述する)。次に、OS 領域 2 1 0 の中に、以下の変数およびテーブルの領域を確保し、所要の初期値に初期化する (s 3 0 2 ~ s 3 0 6)。プロセス数を示す変数 N\_\_p r o c を確保し、初期値を 0 とする (s 3 0 2)。実行中のプロセス番号を示す変数 C\_\_p r o c を確保し、初期値を 0 とする (s 3 0 3)。

#### 【0 0 5 9】

次に、図 8 a に示した形式のプロセス管理テーブル 7 0 0 の領域を確保し初期化する (s 3 0 4)。なおプロセス番号 0 は、OS 自身を示すものとする。次に、図 8 b に示した形式の状態保存テーブル 7 5 0 の領域を確保する (初期化不要) (s 3 0 5)。更に、キャッシュ管理テーブル領域を確保して内容を初期化する (s 3 0 6)。この詳細は後述する。

#### 【0 0 6 0】

以上の準備が完了した時点で、タイマ 1 0 1 0 からの割込を許可し (s 3 0 7)、OS の実行を休止する。以後 OS は割込の都度起動される。

#### 【0 0 6 1】

次にタイマ 1 0 1 0 から割込が生じた場合の動作を説明する。この動作を行うプログラムは、割込アドレス 2 0 1 以降に格納される。

#### 【0 0 6 2】

割込みが生じた場合に OS が行なう動作は、2 つに大別される。第一の動作は、オペレータからの操作入力に従って所定のアプリケーションプログラムを実行する事であり、第二の動作は、複数のプログラム単位 (プロセス) が時分割で実行されてる状況において、プロセスを現在実行中のものから他のものへ切換える事 (プロセス・スイッチ) である。以上の詳細を、図 4 のフローチャートを用いて説明する。

#### 【0 0 6 3】

割込が生じた場合、OS はまず現在の CPU の状態を状態保存テーブル 7 5 0 内の退避領域 7 6 0 に記憶する (s 3 1 1)。これは割込の直前まで実行されていたプロセスを、OS の動作が完了した後に元の状態で再開できるようにするためである。記憶する内容は、CPU の種類により異なるが、割込時に実行中のア

ドレスを示すPC値752、CPUのフラグ状態753、CPU内のレジスタの値754、755、……、等である。

【0064】

次にOSは、入力装置1006にオペレータからの操作が入力されたかどうかを調べ(s312)、その結果に応じて上記第一の動作(s321～s327)と第二の動作(s331～s337)とに分けて動作を行なう。

【0065】

第一の動作の場合には、OSはまず入力装置1006から実行すべきアプリケーションプログラムの名称を入力する(s321)。次に補助記憶装置1005から当該プログラムの属性情報231を取得し(s322)、この情報と前記各種テーブルの内容を勘案して、プロセスの実行に必要な資源であるプログラム領域、作業領域などを、資源割当手段999により割当てする(s323)。更に、割当の結果に従ってプロセス管理テーブルを更新する(s324)。以上の更新が完了した後、プロセス数を示す変数N\_procを1つ増やし(s325)、この新しいプロセスに対する状態保存テーブルを初期化する(s326)。すなわち、テーブル内のPC値をプロセスの実行開始アドレスとなるプログラム領域開始番地703の内容 + エントリアドレス902の値(後述)とし、他の値は適当な初期値(たとえば0)とする。

【0066】

最後にプログラム221をs323で定めた主記憶アドレスにロードする(s327)。これにより、次にプロセス番号3に実行の順番が回ってきた時に、同プロセスを初期状態から実行する事が可能となる。これ以後の処理は、後に記すの第二の動作に合流する。

【0067】

なお以上においては簡略のため、補助記憶装置1005に記憶されたアプリケーションプログラム221、222、……、は、それぞれが1つのプロセスとして実行されるものとした。

【0068】

第二の動作では、以下のs331～s337によりプロセスの切換作業を行な

う。

#### 【0069】

まずプロセス数を示す変数N\_\_procを参照し、その値により下記のいずれかの処理へ分岐する(s331)。変数N\_\_procが0の場合は、何もせずにOSの実行を休止し、以後の割込を待つ。変数N\_\_procが1の場合は、CPUの状態を退避領域760の内容に戻し(s332)、プロセス番号C\_\_procの実行に移る。変数N\_\_procが2以上の場合は以下のs333～s337により、実行するプロセスを現在のC\_\_proc番から次の番号へ切換える。

#### 【0070】

まず退避領域760の内容を、状態保存テーブル中の当該プロセス用領域770に複写し(s333)、実行すべきプロセスの番号C\_\_procを1つ増やす(s334)。但しC\_\_procがN\_\_procを超えた場合は番号を1番に戻す(s335～s336)。

#### 【0071】

次に、CPUの状態を状態保存テーブル中に保存されたC\_\_proc番プロセスの状態に戻し(s337)、プロセス番号C\_\_procの実行に移る。

#### 【0072】

なお実際に用いられるOSでは、上述の機能がもっと複雑となる。たとえば特定のプロセスを優先的に実行させるために、同一のプロセスを2回続けて実行する事がある。また、複数のプロセスで資源の競合が生じている場合(たとえばプロセスAが作成するデータファイルがプロセスBの入力として使われる場合など)には、一方のプロセスを強制的に待たせる等の調整を行なう。更に、オペレータからの操作入力、上例で示した様なアプリケーション実行の操作に限らず、例えば現在実行中のプロセスを停止させる操作などが考えられ、相応の機能がOSに要求される。しかしながらこれらは、本発明の実施に影響を与えるものではないので、ここでは割愛する。

#### 【0073】

以上の説明において、キャッシュ管理テーブル800、及びs323で用いられる割当手段999は、ハードウェアの構成に応じていくつかの構成が考えられ

る。これらを以下に順次説明する。

【0074】

まず第1の実施例として、キャッシュメモリが図5に示した「ダイレクトマッピング方式」の構成である場合について説明する。

【0075】

この場合スラッシングを防ぐには、複数のプロセス各々の主要部がキャッシュメモリの同一アドレスを使用しないようにプロセスを主記憶上に配置すれば良い。このためには、アプリケーションプログラムの属性情報231、232、……、から各々の主要部位置を取得し、それらから式1により定まるアドレスが相互に異なる様に、プロセスをロードする主記憶アドレスを決定すれば良い。

【0076】

この場合、属性情報として必要な情報は図11aの形式となる。即ち、属性情報231には、プログラム名称900、プログラムサイズ901、エントリアドレス902、作業領域サイズ903、に加えて、主要部位置910、主要部サイズ911が記録される。エントリアドレス902および主要部位置910は、当該プログラム先頭からの相対バイト位置で示す。

【0077】

またキャッシュ管理テーブルは、キャッシュメモリの各アドレスが現在どのプロセスにより使用されているか、或いは未使用であるか、を図9aの形式で記憶する。即ち、キャッシュ管理テーブル800は、キャッシュ番地810と使用するプロセス番号820との対応表である。あるアドレスが未使用の場合は、プロセスが存在しない事を示す値-1をプロセス番号820に記しておく。OSの準備動作におけるキャッシュ管理テーブル作成(s306)においては、OSが自ら使用するアドレスに0を、他の全てのアドレスに-1を記しておけば良い。なお図9aでは簡略化のため、キャッシュメモリのアドレスは128(\$80)バイトずつまとめて扱うものとした。キャッシュ管理テーブル800の実現方法はこれに限らず、たとえば各プロセス番号に対応するキャッシュメモリのアドレスの範囲を記憶する方法でも良い。

## 【0078】

以上の属性情報及びキャッシュ管理テーブル、並びに前述の資源管理テーブル700を用いて、資源割当手段999は図1のフローチャートに従って動作する。

## 【0079】

まず、新たなプロセスのプロセス番号を、実行中のプロセス数に1を加えた値  $N\_proc + 1$  とする (s2101)。これ以降の資源管理テーブル700への書き込みは、プロセス番号  $N\_proc + 1$  を対象に行なう。プログラム名称900をプログラム名の欄702に書き込む (s2102)。作業領域サイズ903より大きな主記憶の空き領域を資源管理テーブル700を用いて探し (s2103)、見つかった領域の開始番地を欄706に、開始番地+作業領域サイズ-1を欄707に書き込む (s2104)。これにより主記憶上に作業領域が確保される。

## 【0080】

プログラムサイズ901+キャッシュサイズ(8KB)より大きな主記憶の空き領域を資源管理テーブル700を用いて探す (s2105)。ここでキャッシュサイズ(8KB)を加算する理由は、後のs2108においてロードするアドレスをずらす余裕を設けるためである。

## 【0081】

主要部サイズ911より大きなキャッシュメモリの空き領域をキャッシュ管理テーブル800を用いて探す (s2106)。前記s2106で見つかった領域の開始番地から主要部サイズ分のキャッシュ番地810に対応するプロセス番号欄820にプロセス番号  $N\_proc + 1$  を書き込む (s2107)。これにより同プロセスの主要部に対するキャッシュメモリが確保される。

## 【0082】

プログラムをロードする主記憶アドレスLを以下の式3に従って定める (s2108)。

## 【0083】

$$L = (C - P - F) \bmod S + F \quad \text{[式3]}$$

但し C: s 2 1 0 6 で見つかったキャッシュメモリ領域の先頭番地

P: 主要部位置 9 1 0 の値

S: キャッシュメモリのサイズ

F: s 2 1 0 5 で見つかった主記憶空き領域の先頭番地

本ステップにより、ロード後に主要部が使用するキャッシュメモリのアドレスが s 2 1 0 6 で見つかったキャッシュメモリの空き領域に一致するようになる。

#### 【 0 0 8 4 】

上記 s 2 1 0 8 による L を欄 7 0 3 に、L + プログラムサイズ - 1 を欄 7 0 4 に、L + エントリアドレスを欄 7 0 5 に、それぞれ書き込む (s 2 1 0 9)。これにより、主記憶上にプログラム領域が確保される。

#### 【 0 0 8 5 】

以上によりプロセス実行に必要な資源が得られたので、資源割当手段 9 9 9 の動作を終了し、図 4 b のフローチャートに戻る。

#### 【 0 0 8 6 】

次に第 2 の実施例として、ハードウェアが図 6 に示した方式の場合について説明する。図 6 は「アプリケーション空間分割方式」と呼ばれる、図 5 を改良したキャッシュメモリの構成例である。

#### 【 0 0 8 7 】

図 6 においては、キャッシュメモリシステムの構成要素や動作は図 5 と同じであるが、エントリアドレスの作り方が異なっている。すなわち、図 5 においてはエントリアドレスは PC の値のうちの第 2 ~ 1 3 b i t を用いていたが、図 6 においてはエントリアドレスの上位 2 ビットに、PC の上位 2 ビット (第 2 3 ~ 2 4 b i t) をそのまま用いる。これに伴い、タグ 1 1 2 はビット数は不変であるが、取出す位置に変更が生じ、PC の第 1 1 ~ 2 2 b i t (全 1 2 ビット) を用いるようになる。この構成により、エントリアドレスは

\$ 0 0 0 ~ \$ 1 F F	.....	ページ 0
\$ 2 0 0 ~ \$ 3 F F	.....	ページ 1
\$ 4 0 0 ~ \$ 5 F F	.....	ページ 2
\$ 6 0 0 ~ \$ 7 F F	.....	ページ 3

の4つに分けられ、各々のページが独立したキャッシュメモリとして機能する。キャッシュメモリの実質的な容量であるライン群153の容量は、図5と図6で同一である。

## 【0088】

主記憶アドレスとキャッシュメモリアドレス（エントリアドレスではなくバイト単位のアドレス）の対応は、図5においては前述の通り

$$A = A \bmod (2^{**} 13) \quad \text{[式1]}$$

となるのに対し、本構成例においては、

$$A = A \bmod (2^{**} 11) + \$200 \times (A \div \$800000)$$

但し  $\div$  は整数部のみを値とするものとする …… [式4]

となる。[式4]の第2項である $\$200 \times (A \div \$800000)$ は、アプリケーションプログラムがどのページを用いるかを、同プログラムに割当られたメモリアドレスに従って決定する部分である。言い換えれば、本方式においては、アプリケーションプログラムのメモリへの割付を決定する事が、そのままキャッシュメモリのページを割当る事になる。

## 【0089】

図6の構成においてスラッシングを防ぐには、各々のプロセスが異なるキャッシュメモリのページを使用する様に、プロセスを主記憶上に配置すれば良い。この場合、各プロセスの主要部に関する情報は必要がなくなり、属性情報として必要な情報は図11bの形式となる。即ち、属性情報231には、プログラム名称900、プログラムサイズ901、エントリアドレス902、作業領域サイズ903があれば良く、図11aに示した第1の実施例の場合よりも簡単になる。

## 【0090】

またキャッシュ管理テーブルは、キャッシュメモリの各ページが現在どのプロセスにより使用されているか、乃至は未使用であるか、を図9bの形式で記憶する。即ち、キャッシュ管理テーブル800は、キャッシュページ811と使用するプロセス番号820との対応表である。あるページが未使用の場合は、プロセスが存在しない事を示す値-1をプロセス番号820に記しておく。OSの準備動作におけるキャッシュ管理テーブル作成(s306)においては、OSが自ら

使用するページに 0 を、他の全てのページに -1 を記しておけば良い。

【0091】

以上の属性情報およびキャッシュ管理テーブル、並びに前述の資源管理テーブル 700 を用いて、資源割当手段 999 は図 12 のフローチャートに従って動作する。

【0092】

まず、新たなプロセスのプロセス番号を、実行中のプロセス数に 1 を加えた値  $N\_proc + 1$  とする (s2201)。これ以降の資源管理テーブル 700 への書き込みは、プロセス番号  $N\_proc + 1$  を対象に行なう。

【0093】

プログラム名称 900 をプログラム名の欄 702 に書き込む (s2202)。作業領域サイズ 903 より大きな主記憶の空き領域を資源管理テーブル 700 を用いて探し (s2203)、見つかった領域の開始番地を欄 706 に、開始番地 + 作業領域サイズ - 1 を欄 707 に書き込む (s2204)。これにより主記憶上に作業領域が確保される。

【0094】

以上の s2201 ~ s2204 は、第 1 の実施例における s2101 ~ s2104 と全く同一である。

【0095】

更に、キャッシュメモリの空きページをキャッシュ管理テーブル 800 を用いて探し (s2205)、上記 s2205 で見つかったページに対応するプロセス番号欄 820 にプロセス番号  $N\_proc + 1$  を書き込む (s2206)。これにより同プロセスに対するキャッシュメモリのページが確保される。

【0096】

プログラムをロードする主記憶アドレス  $L$  を以下の式 5 に従って定める (s2207)。

【0097】

$$L = C * (2^{**} 23) \quad \text{[式 5]}$$

但し  $C$ : s2205 で見つかったキャッシュメモリのページ番号



本ステップにより、ロード後に主要部が使用するキャッシュメモリのページが s 2 2 0 5 で見つかったページに一致するようになる。

【0098】

上記 s 2 2 0 7 による L を欄 7 0 3 に、L + プログラムサイズ - 1 を欄 7 0 4 に、L + エントリアドレスを欄 7 0 5 に、それぞれ書き込む (s 2 2 0 8)。これにより、主記憶上にプログラム領域が確保される。

【0099】

以上によりプロセス実行に必要な資源が得られたので、割当手段 9 9 9 の動作を終了し、図 4 のフローチャートに戻る。

【0100】

なお本実施例においては、1つのプロセスにキャッシュメモリを2ページ割付ける事も可能である。このためには、s 2 2 0 5 において連続した空きページ C'、C' + 1 を見つけておき、s 2 2 0 7 では例えば、

$$L = (C + 1) * (2^{23}) - (\text{プログラムサイズ} \div 2) \quad \text{[式 5']}$$
なる L' をプログラムをロードする主記憶アドレスとする。この結果、プログラムの前半はキャッシュページ C' を、後半は C' + 1 を用いる事になる。

【0101】

但し1つのプロセスが複数のキャッシュページを占有することは、他のプロセスの処理速度を犠牲にする事になるので、複数ページを割付けるかどうかは、プロセスの優先度を考慮して判定する必要がある。このためには図 1 1 c に示す様に、要求キャッシュページ 9 2 0 (又はこれと等価な優先度情報) を属性情報 2 3 1 に予め記録しておき、キャッシュページに空きがあつてかつプロセスが複数ページを要求する場合にのみ式 5' 採用する、とすれば良い。

【0102】

また、以上の説明においては、1つのキャッシュページには高々1つのプロセスしか割付けられないものとしたが、このままでは同時に実行できるプロセスの数がキャッシュページの数に制限されてしまい、効果的でない。そこで、本実施例と第 1 の実施例とを併用した、以下の様な方法が考えられる。

## 【0103】

属性情報 231 にはプロセスの優先度情報 930 を予め記録しておく(図 11 d)。優先度情報は下記 3 段階のいずれかである。

優先度 3 (最優先) あるキャッシュページを排他的に使用する。

優先度 2 (優先) キャッシュページを他プロセスと共用することがある。但し各々の主要部は、キャッシュアドレスを他と共有しない。

優先度 1 (普通) キャッシュページを他プロセスと共用することがあり、主要部が同一のキャッシュアドレスとなる可能性もある(スラッシングの危険を認める)。

## 【0104】

割付手段 999 は優先度情報 930 を参照して、優先度 3 のプロセスに対しては第 2 の実施例の方法で、空いているキャッシュページを割付る。優先度 2 のプロセスに対しては、キャッシュページに空きがない場合は、適当なキャッシュページ(但し優先度 3 のプロセスが使用していないページ)を選択して、第 1 の実施例の方法によりキャッシュアドレスを割付ける。優先度 1 のプロセスに対しては、第 1 の実施例の方法を試みるが、キャッシュアドレスの割付が不可能な場合であってもそのままプロセスをロードし実行する。

## 【0105】

以上を実施する為のキャッシュ管理テーブル 800 としては、図 10 に示したものを使用する。図 10 は図 9 a と図 9 b を組み合わせたテーブルであり、キャッシュページ番号 811 と各ページ内のキャッシュアドレス(ページ先頭アドレスからの相対値) 810 からなる 2 次元の表である。表の各欄 821 には、当該ページの当該番地を使用するプロセスの番号が記される。値 -1 は未使用を示す。排他使用欄 830 は、当該ページを排他的に使用するプロセスが存在するかどうかを表す。存在する場合はそのプロセス番号を示し、値 -1 は複数のプロセスが当該ページを共用している事を、値 -2 は当該ページが全く使用されていない事を示す。本テーブルの初期化(s306)は、排他使用欄 830 を全て -2、各欄 821 を全て -1 とした後、OS(プロセス番号 0)の使用部分を書き込めば良い。

## 【0106】

更に第3の実施例として、ハードウェアが図7に示した方式の場合について説明する。図7は「キャッシュページレジスタ方式」と呼ばれる、図5を改良したキャッシュメモリの構成例である。

## 【0107】

図7においては、キャッシュメモリシステムの構成要素や動作は図5と同じであるが、エントリアドレスの作り方が異なっている。すなわち、図5においてはエントリアドレスはPCの値のうちの第2～12bitを用いていたが、図7においては2ビットのレジスタであるキャッシュページレジスタ160が設けられ、エントリアドレスの上位2ビットとして用いられる。キャッシュページレジスタの内容は、適当なCPU命令によりソフトウェア的に変更されるものとする。これに伴い、タグ112が2ビット増加し、PCの第11～24bit（全14ビット）を用いる様になっている。

## 【0108】

この構成により、エントリアドレスは図6と同様に、

\$ 000	～	\$ 1FF	.....	ページ0
\$ 200	～	\$ 3FF	.....	ページ1
\$ 400	～	\$ 5FF	.....	ページ2
\$ 600	～	\$ 7FF	.....	ページ3

の4つに分けられ、各々のページが独立したキャッシュメモリとして機能する。従ってスラッシングを防ぐ方策も、第2の実施例とほぼ同一ある。異なるのは、図6においては主記憶アドレスの違いによってキャッシュページが自動的に切換えられていたのに対し、図7においてはOSが明示的にキャッシュページレジスタを書換える必要が有る、という点である。すなわち図14に示す通り、図4のプロセス切換手順におけるステップ337の後に、キャッシュページレジスタ160をキャッシュ管理テーブル800に従って書き換える操作（s338）が加わる。

## 【0109】

本実施例では、属性情報231およびキャッシュ管理テーブル800は第2の

実施例と同一であり、各々図 11b および図 9b に記したものをを用いる。

【0110】

資源割当手段 999 は図 13 のフローチャートに従って動作する。図 13 の s2301～s2308 は、図 12 の s2201～s2208 と s2307 を除いて同一である。s2307 では、プログラムサイズ 901 より大きな主記憶の空き領域を資源割当テーブル 700 を用いて探す事により、プログラムをロードする主記憶アドレス L を定める。

【0111】

本実施例も第 2 の実施例と同様に、第 1 の実施例で示した方法との併用が可能である。

【0112】

以上、本発明の実施例を 3 通り説明した。

【0113】

これらの実施例においては、属性情報 231 は外部記憶装置 1005 から読み込むものとしたが、プロセス起動を要求する際のパラメータとして属性情報を OS に与える方法でも良い。

【0114】

また、資源割当手段 999 が割当を失敗した場合の動作については特に述べなかったが、表示装置 1007 にメッセージを表示してプロセス起動を中止する、等適当な措置をとるものとする。

【0115】

またこれらの実施例は、OS の機能の一部として本発明を利用する形態を説明したが、各実施例で示した機能を有するソフトウェアを既存の OS に外付けする形態で、本発明を実施しても良い。

【0116】

【発明の効果】

以上の様に本発明によれば、スラッシングの生じにくいマルチプロセス環境を提供でき、ハードウェアの性能を十分に生かすことが可能な資源の割当方法及び OS 及び前記 OS により動作するコンピュータシステムを提供する事が可能にな

る。

【図面の簡単な説明】

【図 1】

本発明の第 1 の実施例における資源割当手段 9 9 9 の動作を示すフローチャート。

【図 2】

本発明の実施対象となるコンピュータシステムの全体構成を示すブロック図。

【図 3】

OS の準備動作を説明するフローチャート。

【図 4】

OS のタイマから割込が生じた場合の動作を説明するフローチャート。

【図 5】

キャッシュメモリシステムの第 1 の構成（ダイレクトマッピング方式）を示す回路図。

【図 6】

キャッシュメモリシステムの第 2 の構成（アプリケーション空間分割方式）を示す回路図。

【図 7】

キャッシュメモリシステムの第 3 の構成（キャッシュページレジスタ方式）を示す回路図。

【図 8】

OS がマルチプロセスの管理に使用する資源割当テーブルと状態保存テーブルの図。

【図 9】

本発明における第 1 の実施例と、第 2 又は第 3 の実施例におけるキャッシュ管理テーブルの構成を示す図。

【図 1 0】

第 2 又は第 3 の実施例を第 1 の実施例と併用する場合の構成を示す図。

【図 1 1】

本発明における第 1 の実施例と、第 2 又は第 3 の実施例等における属性情報の構成を示す図。

【図 1 2】

第 2 の実施例における資源割当手段 9 9 9 の動作を示すフローチャート。

【図 1 3】

第 3 の実施例における資源割当手段 9 9 9 の動作を示すフローチャート。

【図 1 4】

第 3 の実施例における O S の動作の違いを説明するフローチャート。

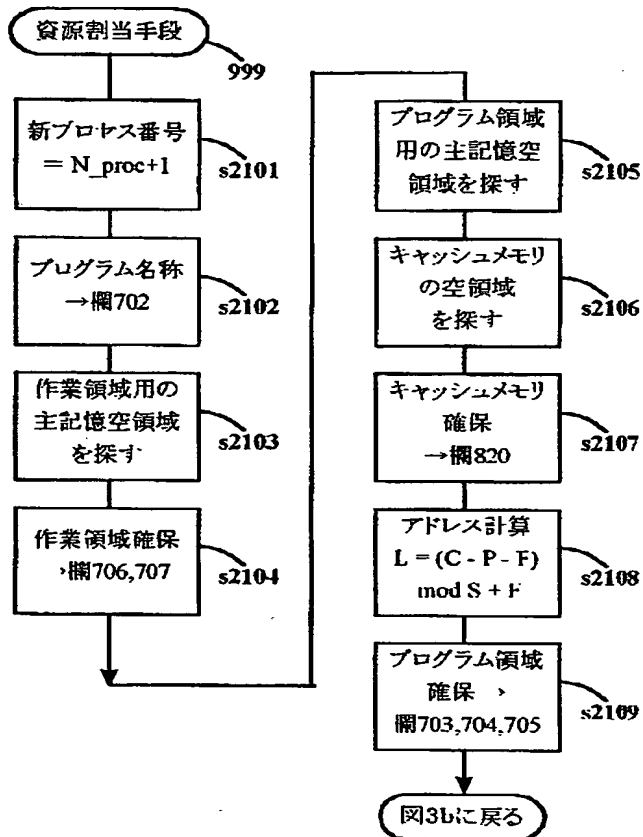
【符号の説明】

1 2 3 : 論理積回路      1 2 4 : 論理否定回路      9 9 9 : 資源割当手段。

【書類名】 図面

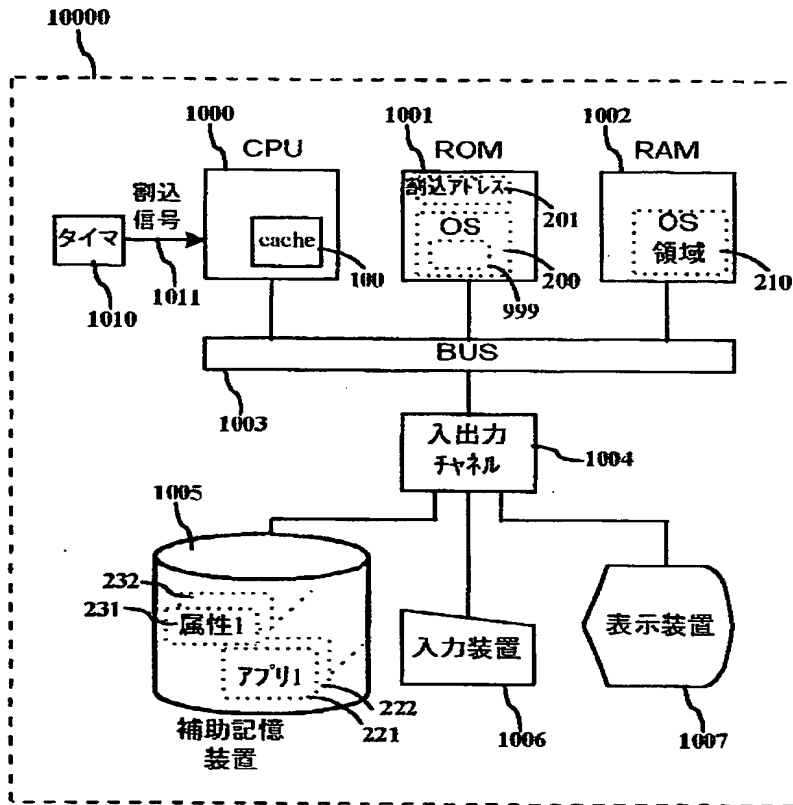
【図 1】

図 1



【図 2】

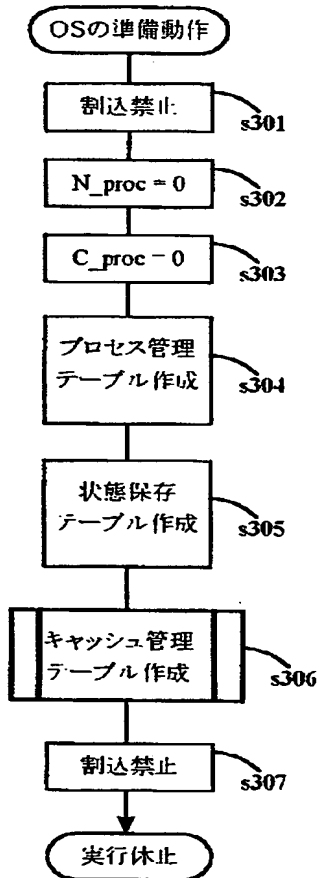
図2





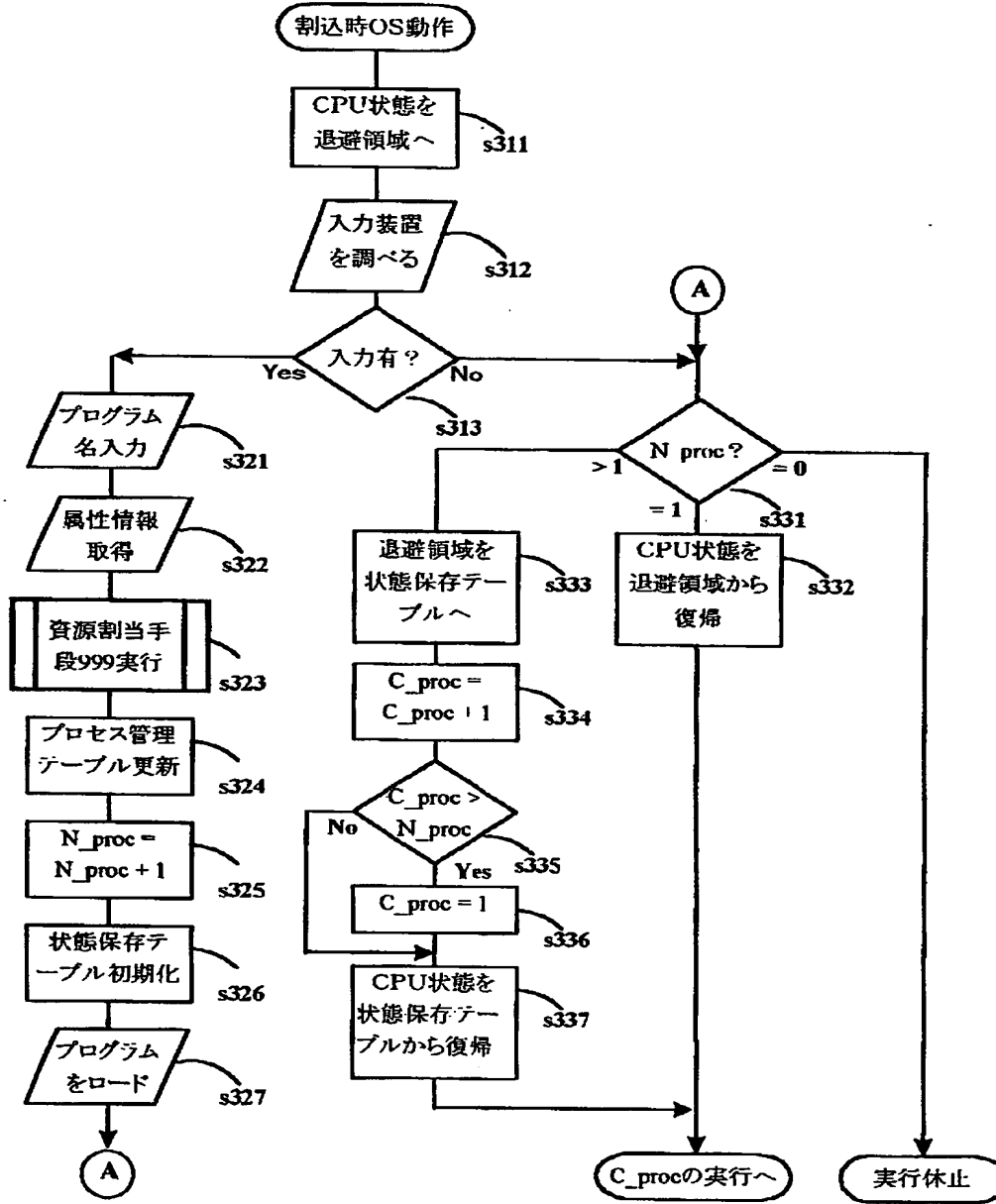
【図 3】

図3



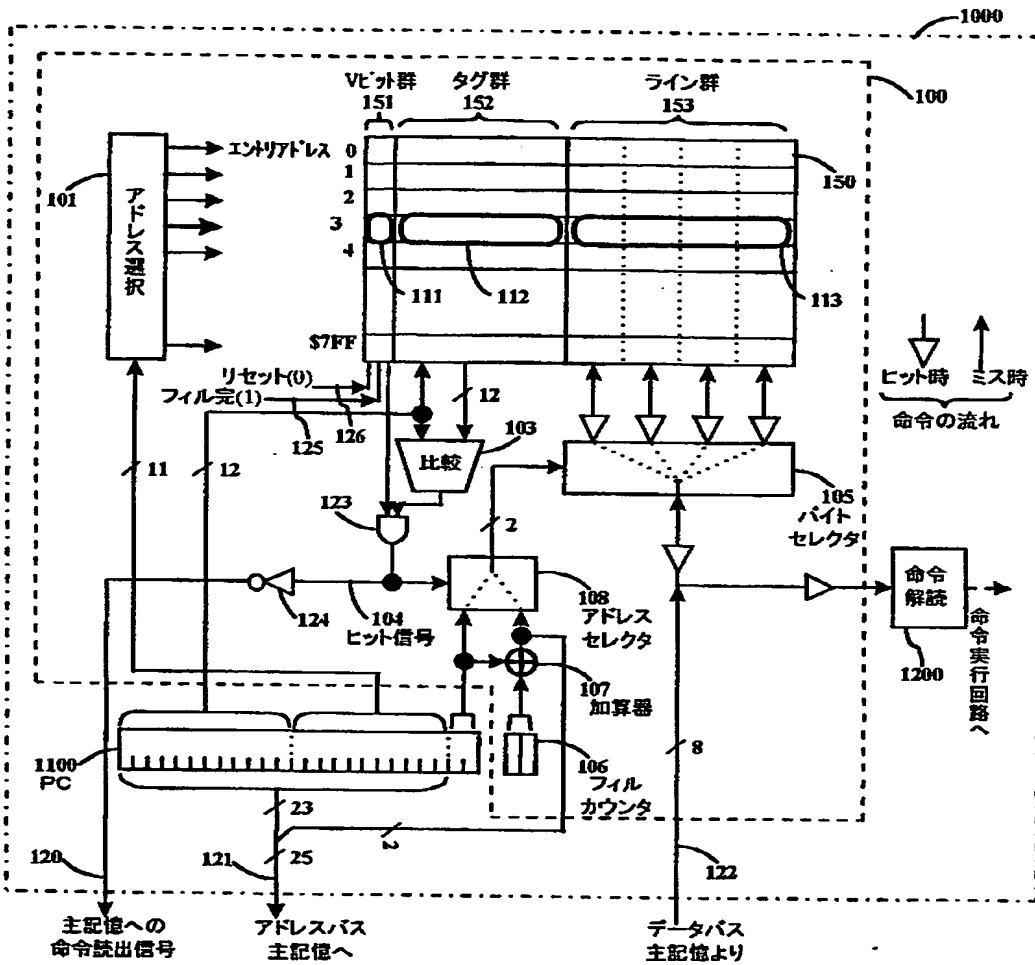
【図 4】

図4



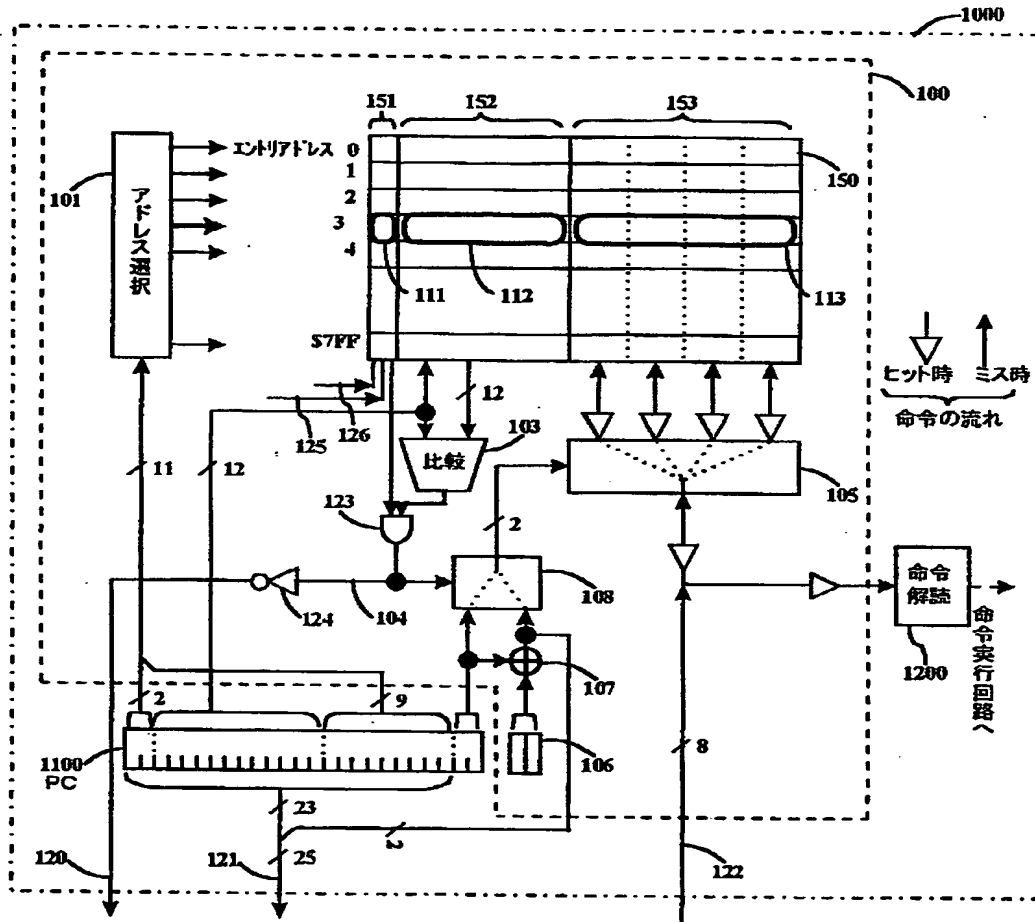
【図 5】

図5

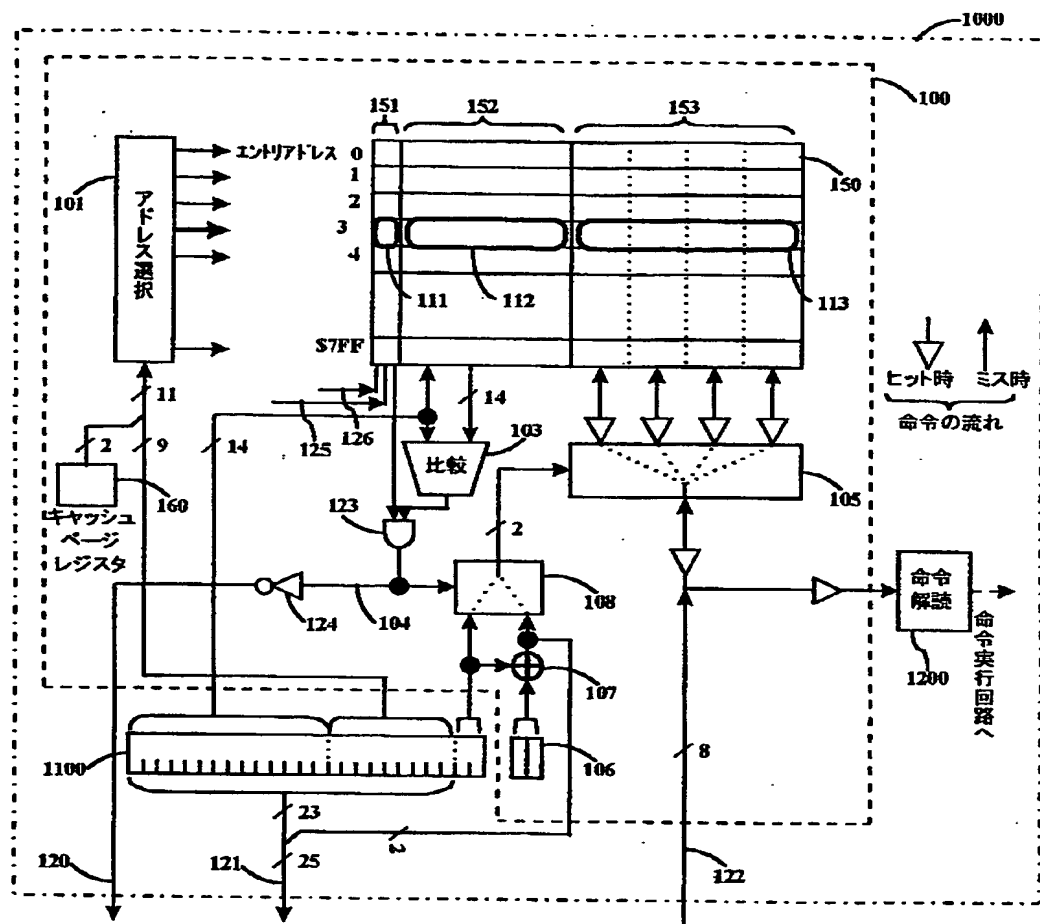


【図 6】

図6



**图7**



【図 8】

図8  
a

701 プロセス番号	702 プログラム名	703 プログラム領域 開始番地	704 プログラム領域 終了番地	705 実行開始番地	706 作業領域 開始番地	707 作業領域 終了番地
0	OS	0	\$1FFFFFF	\$100000	\$200000	\$2FFFFFF
1						
2						
3						
≡	≡	≡	≡	≡	≡	≡

b

751 プロセス番号	752 PC値	753 フラグ状態	754 レジスタ1	755 レジスタ2	
0(OS)					
1					
2					
3					
≡	≡	≡	≡	≡	≡

【図 9】

a

810	820
キャッシュ番地	プロセス番号
\$000-\$07F	0
\$080-\$0FF	0
\$100-\$17F	1
\$180-\$1FF	1
\$200-\$27F	-1
\$280-\$2FF	2
\$300-\$37F	2
\$380-\$3FF	2
\$400-\$47F	-1
\$480-\$4FF	-1
\$500-\$57F	-1
\$580-\$5FF	3
\$1F00-\$1F7F	4
\$1F80-\$1FFF	4

図9

b

811	820
キャッシュページ	プロセス番号
0	0
1	-1
2	-1
3	-1

【図 1 0】

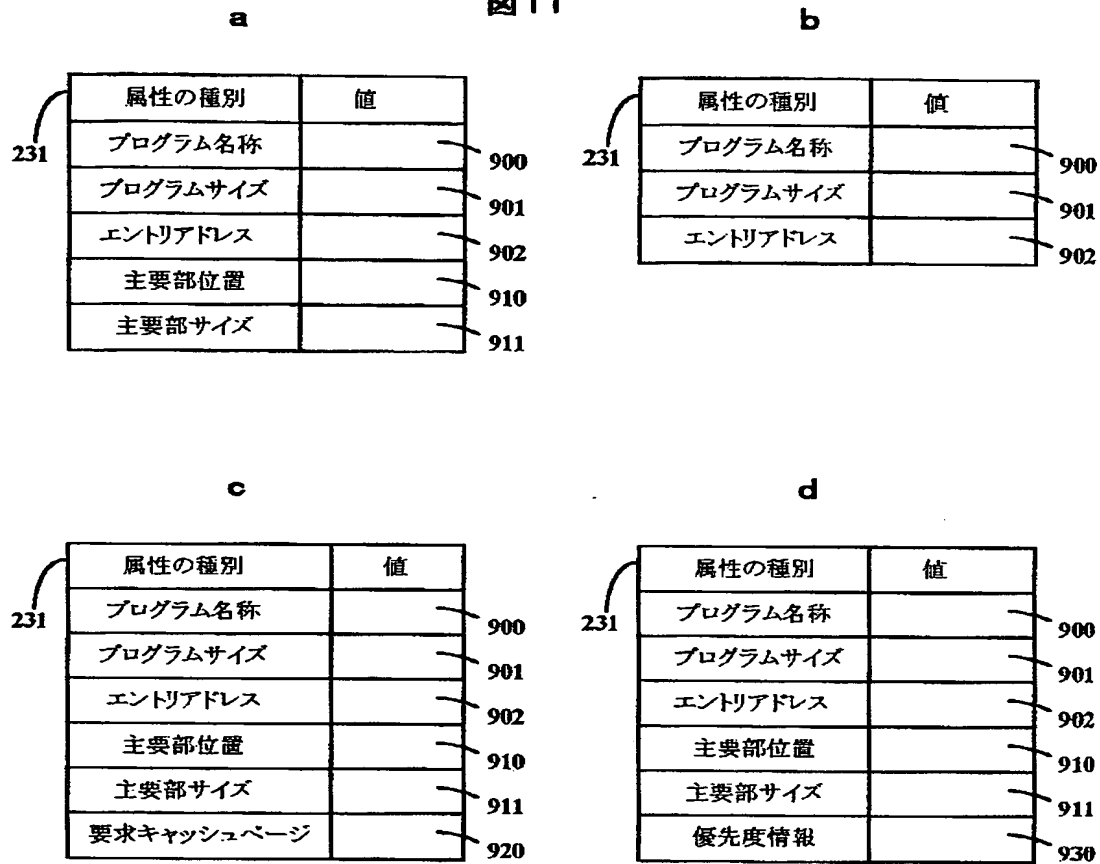
図10

800 キャッシュ ページ 番地	キャッシュ ページ				811
	0	1	2	3	
ページ排他使用	0	-1	-1	-2	830
\$000-\$07F	0	1	3	-1	
\$080-\$0FF	0	1	3	-1	821
\$100-\$17F	0	1	3	-1	
\$180-\$1FF	0	1	-1	-1	
\$200-\$27F	0	-1	-1	-1	
\$280-\$2FF	0	2	5	-1	
\$300-\$37F	0	2	-1	-1	
\$380-\$3FF	0	2	-1	-1	
\$400-\$47F	0	-1	-1	-1	
\$700-\$77F	0	4		-1	810
\$780-\$7FF	0	4		-1	



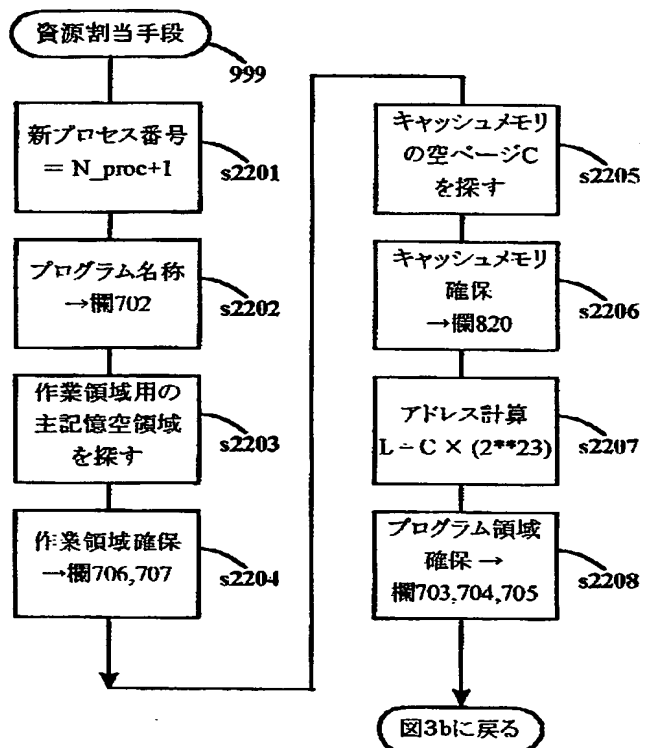
【図 1 1】

図 11



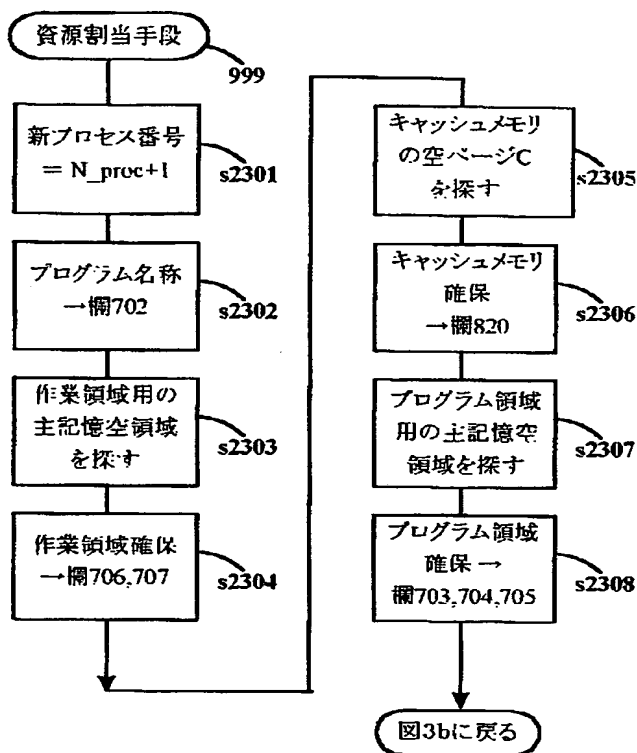
【図 1 2】

図12



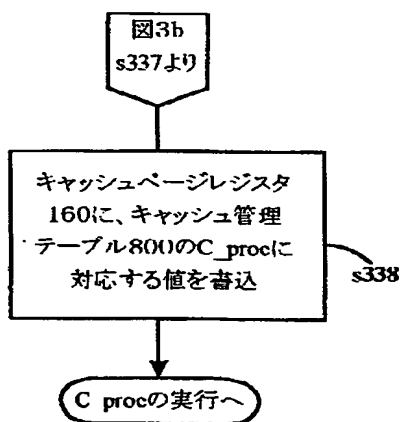
【図 1 3】

図 13



【図 1 4】

図 14



【書類名】 要約書

【要約】

【課題】 マイクロプロセッサにおけるマルチプロセス処理の性能を保証する為に、キャッシュのスラッシングを最小限に押さえる様にプロセスを配置する方法を提供する。

【解決手段】 OS内にキャッシュの使用状態を制御するための管理テーブルを設け、OSはプロセス起動時に、最頻実行部分（主要部）の位置情報を当該プロセスから受取り、管理テーブルとの照合により、主要部に対応するキャッシュアドレスが既存のプロセスとの間で重複しない様に、プロセスのロードアドレスを調整する。調整においてはキャッシュメモリの量、構成方式、プロセスの優先度を勘案して、高優先プロセスが優先的にキャッシュを使用できるようにしたコンピュータシステム。

【選択図】 図 1

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 5 1 0 8]

1. 変更年月日 1 9 9 0 年 8 月 3 1 日  
[変更理由] 新規登録  
住 所 東京都千代田区神田駿河台 4 丁目 6 番地  
氏 名 株式会社日立製作所